

Relational Calculus

R&G, Chapter 4

Relational Calculus

- Comes in two flavors: *Tuple relational calculus* (TRC) and *Domain relational calculus* (DRC).
- Calculus has *variables, constants, comparison ops, logical connectives* and *quantifiers*.
 - *TRC*: Variables range over (i.e., get bound to) *tuples*.
 - Like SQL.
 - *DRC*: Variables range over *domain elements* (= field values).
 - Like Query-By-Example (QBE)
 - Both TRC and DRC are simple subsets of first-order logic.
 - We'll focus on TRC here
- Expressions in the calculus are called *formulas*.
- Answer tuple is an assignment of constants to variables that make the formula evaluate to *true*.

Tuple Relational Calculus

- **Query** has the form: $\{T \mid p(T)\}$
 - $p(T)$ denotes a formula in which tuple variable T appears.
- **Answer** is the set of all tuples T for which the *formula* $p(T)$ evaluates to *true*.
- **Formula** is recursively defined:
 - ❖ start with simple *atomic formulas* (get tuples from relations or make comparisons of values)
 - ❖ build bigger and better formulas using the *logical connectives*.

TRC Formulas

- **An Atomic formula is one of the following:**

$R \in Rel$

$R.a \text{ op } S.b$

$R.a \text{ op } constant$

op is one of $<, >, =, \leq, \geq, \neq$

- **A formula can be:**

– an atomic formula

– $\neg p, p \wedge q, p \vee q$ where p and q are formulas

– $\exists R(p(R))$ where variable R is a tuple variable

– $\forall R(p(R))$ where variable R is a tuple variable

Free and Bound Variables

- The use of **quantifiers** $\exists X$ and $\forall X$ in a formula is said to **bind** X in the formula.
 - A variable that is **not bound** is **free**.
- **Let us revisit the definition of a query:**
 - $\{T \mid \rho(T)\}$
- **There is an important restriction**
 - the variable T that appears to the left of `|' must be the **only** free variable in the formula $\rho(T)$.
 - in other words, all other tuple variables must be bound using a quantifier.

Selection and Projection

- **Find all sailors with rating above 7**

$$\{S \mid S \in \textit{Sailors} \wedge S.\textit{rating} > 7\}$$

- Modify this query to answer: Find sailors who are older than 18 or have a rating under 9, and are called 'Bob'.

- **Find names and ages of sailors with rating above 7.**

$$\{S \mid \exists S1 \in \textit{Sailors} (S1.\textit{rating} > 7 \\ \wedge S.\textit{sname} = S1.\textit{sname} \\ \wedge S.\textit{age} = S1.\textit{age})\}$$

- Note: S is a tuple variable of 2 fields (i.e. $\{S\}$ is a **projection of *Sailors***)

- only 2 fields are ever mentioned **and** S is never used to range over any relations in the query.

Joins

Find sailors rated > 7 who've reserved boat #103

$$\{S \mid S \in \text{Sailors} \wedge S.\text{rating} > 7 \wedge \\ \exists R(R \in \text{Reserves} \wedge R.\text{sid} = S.\text{sid} \\ \wedge R.\text{bid} = 103)\}$$

Note the use of \exists to find a tuple in Reserves that `joins with' the Sailors tuple under consideration.

Joins (continued)

$$\{S \mid S \in \text{Sailors} \wedge S.\text{rating} > 7 \wedge \\ \exists R(R \in \text{Reserves} \wedge R.\text{sid} = S.\text{sid} \\ \wedge \exists B(B \in \text{Boats} \wedge B.\text{bid} = R.\text{bid} \\ \wedge B.\text{color} = \text{'red'}))\}$$

Find sailors rated > 7 who've reserved a red boat

- **Observe how the parentheses control the scope of each quantifier's binding.**
- **This may look cumbersome, but it's not so different from SQL!**

Division (makes more sense here???)

Find sailors who've reserved all boats

(hint, use \forall)

$$\{S \mid S \in \text{Sailors} \wedge \\ \forall B \in \text{Boats} (\exists R \in \text{Reserves} \\ (S.\text{sid} = R.\text{sid} \\ \wedge B.\text{bid} = R.\text{bid}))\}$$

- Find all sailors ***S*** such that for all tuples ***B*** in Boats there is a tuple in Reserves showing that sailor ***S*** has reserved ***B***.

Division – a trickier example...

Find sailors who've reserved all **Red** boats

$$\{S \mid S \in \text{Sailors} \wedge \\ \forall B \in \text{Boats} (B.\text{color} = \text{'red'} \Rightarrow \\ \exists R (R \in \text{Reserves} \wedge S.\text{sid} = R.\text{sid} \\ \wedge B.\text{bid} = R.\text{bid}))\}$$

Alternatively...

$$\{S \mid S \in \text{Sailors} \wedge \\ \forall B \in \text{Boats} (B.\text{color} \neq \text{'red'} \vee \\ \exists R (R \in \text{Reserves} \wedge S.\text{sid} = R.\text{sid} \\ \wedge B.\text{bid} = R.\text{bid}))\}$$

$a \Rightarrow b$ is the same as $\neg a \vee b$

		b	
		T	F
a	T	T	F
	F	T	T

- **If a is true, b must be true!**
 - If a is true and b is false, the implication evaluates to false.
- **If a is not true, we don't care about b**
 - The expression is always true.

Unsafe Queries, Expressive Power

- \exists syntactically correct calculus queries that have an infinite number of answers! **Unsafe** queries.
 - e.g., $\{S \mid \neg(S \in Sailors)\}$
 - Solution???? Don't do that!
- **Expressive Power (Theorem due to Codd):**
 - every query that can be expressed in relational algebra can be expressed as a *safe* query in DRC / TRC; the converse is also true.
- **Relational Completeness**: Query language (e.g., SQL) can express every query that is expressible in relational algebra/calculus. (actually, SQL is more powerful, as we will see...)

Summary

- **The relational model has rigorously defined query languages — simple and powerful.**
- **Relational algebra is more operational**
 - useful as internal representation for query evaluation plans.
- **Relational calculus is non-operational**
 - users define queries in terms of what they want, not in terms of how to compute it. (*Declarative*)
- **Several ways of expressing a given query**
 - a *query optimizer* should choose the most efficient version.
- **Algebra and safe calculus have same *expressive power***
 - leads to the notion of *relational completeness*.

Addendum: Use of \forall

- $\forall x (P(x))$ - is only true if $P(x)$ is true for *every* x in the universe
- Usually:
 $\forall x ((x \in \text{Boats}) \Rightarrow (x.\text{color} = \text{"Red"}))$
- \Rightarrow logical implication,
 $a \Rightarrow b$ means that if a is true, b must be true
 $a \Rightarrow b$ is the same as $\neg a \vee b$

Find sailors who've reserved all boats

$$\{S \mid S \in \text{Sailors} \wedge \\ \forall B((B \in \text{Boats}) \Rightarrow \\ \exists R(R \in \text{Reserves} \wedge S.\text{sid} = R.\text{sid} \\ \wedge B.\text{bid} = R.\text{bid}))\}$$

- **Find all sailors S such that for each tuple B either it is not a tuple in Boats or there is a tuple in Reserves showing that sailor S has reserved it.**

$$\{S \mid S \in \text{Sailors} \wedge \\ \forall B(\neg(B \in \text{Boats}) \vee \\ \exists R(R \in \text{Reserves} \wedge S.\text{sid} = R.\text{sid} \\ \wedge B.\text{bid} = R.\text{bid}))\}$$

... reserved all red boats

$$\{S \mid S \in \text{Sailors} \wedge \\ \forall B((B \in \text{Boats} \wedge B.\text{color} = \text{"red"}) \Rightarrow \\ \exists R(R \in \text{Reserves} \wedge S.\text{sid} = R.\text{sid} \\ \wedge B.\text{bid} = R.\text{bid}))\}$$

- **Find all sailors S such that for each tuple B either it is not a tuple in Boats or there is a tuple in Reserves showing that sailor S has reserved it.**

$$\{S \mid S \in \text{Sailors} \wedge \\ \forall B(\neg(B \in \text{Boats}) \vee (B.\text{color} \neq \text{"red"}) \vee \\ \exists R(R \in \text{Reserves} \wedge S.\text{sid} = R.\text{sid} \\ \wedge B.\text{bid} = R.\text{bid}))\}$$